Варіант 7

Мета роботи:

засвоїти методи розв'язання диференціальних рівнянь першого роду.

Завдання:

- Методами Рунге-Кутта четвертого порядку та Адамса четвертого порядку розв'язати задачу Коші

$$y' = f(x, y);$$
$$y(x_o) = y_o;$$

на відрізку[x_o; x_o+ 1] з кроком h.

На початку інтервалу у необхідній кількості точок значення для методу Адамса визначити методом Рунге-Кутта четвертого порядку.

- Дослідити залежність помилки обох методів від кроку дискретизації h. Для цього розв'яжіть задачу Коші, розбиваючи інтервал розв'язання [x_o; x_o+ 1] на різні кількості відрізків, щоразу фіксуючи максимальну помилку.

Допрограмний етап

$$y' = \frac{3y - 2x + 1}{3x + 3}$$

$$\begin{cases} x = v - 1 \\ y = u - 1 \end{cases}$$

$$u' = \frac{3u - 3 - 2v + 2 + 1}{3v - 3 + 3}$$

$$u' = \frac{3u/v - 2}{3}$$

$$z = u/v \Rightarrow u' = z + z'v$$

$$z + \frac{dz}{dv}v = \frac{3z - 2}{3}$$

$$\frac{dz}{dv}v = \frac{-2}{3}$$

$$3\,dz = -2\frac{dv}{v}$$

$$3z = -2\ln v + C$$

$$3\frac{u}{v} + 2\ln v = C$$

$$3\frac{y+1}{x+1} + 2\ln(x+1) = C$$

Звідси отримуємо, що: y = ((1.0/3)*(x+1)*(C-2*np.log(x+1))-1).

Беремо x0 = 0, y0=0. Тоді C = 3.

Текст програми (одразу з результатами):

```
import numpy as np
import matplotlib.pyplot as plt
import sympy as sp
%matplotlib inline
#defining the function
f = lambda x,y: (3*y-2*x+1)/(3*x+3)
x0 = 0
h = 0.01
x = np.arange(x0, x0+1+h, step=h)
C = 3
y = lambda x: ((1.0/3)*(x+1)*(C-2*np.log(x+1))-1)
#But we also can not just express y via x, but solve our f_result with help of sympy.solve function
l = sp.Symbol('lambda')
f_result=lambda x, y: 2*sp.log(1+x)+3*(y+1)/(x+1)-C
#And so we can find the result in any point x0 with help of two methods
j=0
y_y=[]
for i in x:
    y_y.append(map(sp.re, sp.solve(f_result(i,l), l)))
    print "For x = {x}: sp.solve -- {y_s} ; lambda-function -- {y_l}".format(x=i,y_s=y_y[j][0],
y_l=y(i))
    j+=1
```

```
For x = 0.0: sp.solve -- 0.0 ; lambda-function -- 0.0
For x = 0.01: sp.solve -- 0.00330011055886682 ; lambda-function -- 0.00330011055887
For x = 0.02: sp.solve -- 0.00653421343859784 ; lambda-function -- 0.0065342134386
For x = 0.03: sp.solve -- 0.00970295579413943 ; lambda-function -- 0.00970295579414
For x = 0.04: sp.solve -- 0.0128069722137251 ; lambda-function -- 0.0128069722137
For x = 0.05: sp.solve -- 0.0158468850813976 ; lambda-function -- 0.0158468850814
For x = 0.06: sp.solve -- 0.0188233049257238 ; lambda-function -- 0.0188233049257
For x = 0.07: sp.solve -- 0.0217368307553455 ; lambda-function -- 0.0217368307553
For x = 0.08: sp.solve -- 0.0245880503819876 ; lambda-function -- 0.024588050382
For x = 0.09: sp.solve -- 0.0273775407315020 ; lambda-function -- 0.0273775407315
For x = 0.1: sp.solve -- 0.0301058681434951 ; lambda-function -- 0.0301058681435
For x = 0.11: sp.solve -- 0.0327735886600604 ; lambda-function -- 0.0327735886601
For x = 0.12: sp.solve -- 0.0353812483041043 ; lambda-function -- 0.0353812483041
For x = 0.13: sp.solve -- 0.0379293833477325 ; lambda-function -- 0.0379293833477
For x = 0.14: sp.solve -- 0.0404185205711330 ; lambda-function -- 0.0404185205711
For x = 0.15: sp.solve -- 0.0428491775123785 ; lambda-function -- 0.0428491775124
For x = 0.16: sp.solve -- 0.0452218627085356 ; lambda-function -- 0.0452218627085
For x = 0.17: sp.solve -- 0.0475370759284613 ; lambda-function -- 0.0475370759285
For x = 0.18: sp.solve -- 0.0497953083976423 ; lambda-function -- 0.0497953083976
For x = 0.19: sp.solve -- 0.0519970430154057 ; lambda-function -- 0.0519970430154
For x = 0.2: sp.solve -- 0.0541427545648364 ; lambda-function -- 0.0541427545648
For x = 0.21: sp.solve -- 0.0562329099156892 ; lambda-function -- 0.0562329099157
For x = 0.22: sp.solve -- 0.0582679682205991 ; lambda-function -- 0.0582679682206
For x = 0.23: sp.solve -- 0.0602483811048524 ; lambda-function -- 0.0602483811049
For x = 0.24: sp.solve -- 0.0621745928499919 ; lambda-function -- 0.06217459285
For x = 0.25: sp.solve -- 0.0640470405714917 ; lambda-function -- 0.0640470405715
For x = 0.26: sp.solve -- 0.0658661543907553 ; lambda-function -- 0.0658661543908
For x = 0.27: sp.solve -- 0.0676323576016436 ; lambda-function -- 0.0676323576016
For x = 0.28: sp.solve -- 0.0693460668317645 ; lambda-function -- 0.0693460668318
For x = 0.29: sp.solve -- 0.0710076921987204 ; lambda-function -- 0.0710076921987
For x = 0.3: sp.solve -- 0.0726176374615077 ; lambda-function -- 0.0726176374615
For x = 0.31: sp.solve -- 0.0741763001672608 ; lambda-function -- 0.0741763001673
For x = 0.32: sp.solve -- 0.0756840717935139 ; lambda-function -- 0.0756840717935
For x = 0.33: sp.solve -- 0.0771413378861530 ; lambda-function -- 0.0771413378862
For x = 0.34: sp.solve -- 0.0785484781932140 ; lambda-function -- 0.0785484781932
For x = 0.35: sp.solve -- 0.0799058667946955 ; lambda-function -- 0.0799058667947
For x = 0.36: sp.solve -- 0.0812138722285156 ; lambda-function -- 0.0812138722285
```

For x = 0.37: sp.solve -- 0.0824728576127695 ; lambda-function -- 0.0824728576128For x = 0.38: sp.solve -- 0.0836831807644155 ; lambda-function -- 0.0836831807644
For x = 0.39: sp.solve -- 0.0848451943145235 ; lambda-function -- 0.0848451943145
For x = 0.4: sp.solve -- 0.0859592458202016 ; lambda-function -- 0.0859592458202
For x = 0.41: sp.solve -- 0.0870256778733277 ; lambda-function -- 0.0870256778733
For x = 0.42: sp.solve -- 0.0880448282061996 ; lambda-function -- 0.0880448282062
For x = 0.43: sp.solve -- 0.0890170297942019 ; lambda-function -- 0.0890170297942
For x = 0.44: sp.solve -- 0.0899426109556070 ; lambda-function -- 0.0899426109556
For x = 0.45: sp.solve -- 0.0908218954485994 ; lambda-function -- 0.0908218954486
For x = 0.46: sp.solve -- 0.0916552025656283 ; lambda-function -- 0.0916552025656
For x = 0.47: sp.solve -- 0.0924428472251680 ; lambda-function -- 0.0924428472252
For x = 0.48: sp.solve -- 0.0931851400609897 ; lambda-function -- 0.093185140061
For x = 0.49: sp.solve -- 0.0938823875090146 ; lambda-function -- 0.093882387509
For x = 0.5: sp.solve -- 0.0945348918918355 ; lambda-function -- 0.0945348918918
For x = 0.51: sp.solve -- 0.0951429515009882 ; lambda-function -- 0.095142951501
For x = 0.52: sp.solve -- 0.0957068606770391 ; lambda-function -- 0.095706860677
For x = 0.53: sp.solve -- 0.0962269098875687 ; lambda-function -- 0.0962269098876
For x = 0.54: sp.solve -- 0.0967033858031143 ; lambda-function -- 0.0967033858031
For x = 0.55: sp.solve -- 0.0971365713711393 ; lambda-function -- 0.0971365713711
For x = 0.56: sp.solve -- 0.0975267458880967 ; lambda-function -- 0.0975267458881
For x = 0.57: sp.solve -- 0.0978741850696395 ; lambda-function -- 0.0978741850696
For x = 0.58: sp.solve -- 0.0981791611190511 ; lambda-function -- 0.0981791611191
For x = 0.59: sp.solve -- 0.0984419427939314 ; lambda-function -- 0.0984419427939
For x = 0.6: sp.solve -- 0.0986627954712155 ; lambda-function -- 0.0986627954712
For x = 0.61: sp.solve -- 0.0988419812105612 ; lambda-function -- 0.0988419812106
For x = 0.62: sp.solve -- 0.0989797588161640 ; lambda-function -- 0.0989797588162
For x = 0.63: sp.solve -- 0.0990763838970443 ; lambda-function -- 0.099076383897
For x = 0.64: sp.solve -- 0.0991321089258562 ; lambda-function -- 0.0991321089259
For x = 0.65: sp.solve -- 0.0991471832962617 ; lambda-function -- 0.0991471832963
For x = 0.66: sp.solve -- 0.0991218533789129 ; lambda-function -- 0.0991218533789
For x = 0.67: sp.solve -- 0.0990563625760878 ; lambda-function -- 0.0990563625761
For x = 0.68: sp.solve -- 0.0989509513750122 ; lambda-function -- 0.098950951375
For x = 0.69: sp.solve -- 0.0988058573999203 ; lambda-function -- 0.0988058573999
For x = 0.7: sp.solve -- 0.0986213154628735 ; lambda-function -- 0.0986213154629
For x = 0.71: sp.solve -- 0.0983975576133917 ; lambda-function -- 0.0983975576134
For x = 0.72: sp.solve -- 0.0981348131869188 ; lambda-function -- 0.0981348131869
For x = 0.73: sp.solve -- 0.0978333088521604 ; lambda-function -- 0.0978333088522
For x = 0.74: sp.solve -- 0.0974932686573324 ; lambda-function -- 0.0974932686573
For x = 0.75: sp.solve -- 0.0971149140753400 ; lambda-function -- 0.0971149140753
For x = 0.76: sp.solve -- 0.0966984640479293 ; lambda-function -- 0.0966984640479
For x = 0.77: sp.solve -- 0.0962441350288292 ; lambda-function -- 0.0962441350288
For x = 0.78: sp.solve -- 0.0957521410259273 ; lambda-function -- 0.0957521410259
For x = 0.79: sp.solve -- 0.0952226936424884 ; lambda-function -- 0.0952226936425
For x = 0.8: sp.solve -- 0.0946560021174570 ; lambda-function -- 0.0946560021175
For x = 0.81: sp.solve -- 0.0940522733648669 ; lambda-function -- 0.0940522733649
For x = 0.82: sp.solve -- 0.0934117120123727 ; lambda-function -- 0.0934117120124
For x = 0.83: sp.solve -- 0.0927345204389380 ; lambda-function -- 0.0927345204389
For x = 0.84: sp.solve -- 0.0920208988117032 ; lambda-function -- 0.0920208988117
For x = 0.85: sp.solve -- 0.0912710451220457 ; lambda-function -- 0.091271045122
For x = 0.86: sp.solve -- 0.0904851552208634 ; lambda-function -- 0.0904851552209
For x = 0.87: sp.solve -- 0.0896634228531030 ; lambda-function -- 0.0896634228531
For x = 0.88: sp.solve -- 0.0888060396915385 ; lambda-function -- 0.0888060396915
For x = 0.89: sp.solve -- 0.0879131953698459 ; lambda-function -- 0.0879131953698
For x = 0.9: sp.solve -- 0.0869850775149667 ; lambda-function -- 0.086985077515
For x = 0.91: sp.solve -- 0.0860218717787946 ; lambda-function -- 0.0860218717788
For x = 0.92: sp.solve -- 0.0850237618691968 ; lambda-function -- 0.0850237618692
For x = 0.93: sp.solve -- 0.0839909295803919 ; lambda-function -- 0.0839909295804
For x = 0.94: sp.solve -- 0.0829235548226939 ; lambda-function -- 0.0829235548227
For x = 0.95: sp.solve -- 0.0818218156516481 ; lambda-function -- 0.0818218156516
For x = 0.96: sp.solve -- 0.0806858882965635 ; lambda-function -- 0.0806858882966
For x = 0.97: sp.solve -- 0.0795159471884683 ; lambda-function -- 0.0795159471885
For x = 0.98: sp.solve -- 0.0783121649874940 ; lambda-function -- 0.0783121649875
For x = 0.99: sp.solve -- 0.0770747126097081 ; lambda-function -- 0.0770747126097
For x = 1.0: sp.solve -- 0.0758037592534060 ; lambda-function -- 0.0758037592534

RUNGE-KUTTA method

```
y_ = [y_y[0][0]]

def y_next(x_n,x_next,y_n,h=h,f=f):
    F1 = f(x_n,y_n)
    F2 = f(x_n+h/2.,y_n+h/2.*F1)
    F3 = f(x_n+h/2.,y_n+h/2.*F2)
    F4 = f(x_next,y_n+h*F3)
    return y_n+h/6.*(F1+2*F2+2*F3+F4)
for n in xrange(x.size-1):
    y_.append(y_next(x[n], x[n+1], y_[-1]))
print np.abs(y(x)-y_).max()
```

```
1.03623387648355e-10
```

ADAMS method

```
def Adams_n(y,x,h=h,f=f):
    return y[-1]+h/24.*(55*f(x[-1],y[-1]) - 59*f(x[-2],y[-2]) + 37*f(x[-3],y[-3]) — 9*f(x[-4],y[-4]))
y_a = y_[0:4]
for i in range(4,x.size):
    y_a.append(Adams_n(y_a,x[0:i]))
np.abs(np.subtract(y_a,y(x))).max()
```

```
6.04502171930310e-9
```

Results & errors

```
with open('TOTALYFINNALY', 'w') as f:
    for i in range(x.size):
        if i % 1 == 0:
            print "x = %f" % x[i]
            f.write("x = %f\n" % x[i])
            print "y(x) = %f" % y(x[i])
            f.write("y(x) = %f\n" % y(x[i]))
            print "Rugne Kutta: y1(x) = %f" % y_[i]
            f.write("Rugne Kutta: y1(x) = %f\n" % y_[i])
            print "error:\t%r" % np.abs(y(x[i]) - y_[i])
            f.write("error:\t%r\n" % np.abs(y(x[i]) - y_[i]))
            print "Adams: y2(x) = %f" % y_a[i]
            f.write("Adams: y2(x) = %f\n" % y_a[i])
            print "error:\t%r" % np.abs(y(x[i]) - y_a[i])
            f.write("error:\t%r\n" % np.abs(y(x[i]) - y_a[i]))
            print "--------------------------------"
            f.write("--------------------------------")
```

```
x = 0.000000
y(x) = 0.000000
Rugne Kutta: y1(x) = 0.000000
error:  0
Adams: y2(x) = 0.000000
error:  0
--------------------------------
x = 0.010000
y(x) = 0.003300
Rugne Kutta: y1(x) = 0.003300
error:  2.17568194363360e-12
Adams: y2(x) = 0.003300
error:  2.17568194363360e-12
--------------------------------
x = 0.020000
y(x) = 0.006534
Rugne Kutta: y1(x) = 0.006534
```

```
error:  4.28848796751868e-12
Adams: y2(x) = 0.006534
error:  4.28848796751868e-12
--------------------------------
x = 0.030000
y(x) = 0.009703
Rugne Kutta: y1(x) = 0.009703
error:  6.34129207477407e-12
Adams: y2(x) = 0.009703
error:  6.34129207477407e-12
--------------------------------
x = 0.040000
y(x) = 0.012807
Rugne Kutta: y1(x) = 0.012807
error:  8.33709143388539e-12
Adams: y2(x) = 0.012807
error:  1.22722552606258e-10
--------------------------------
x = 0.050000
y(x) = 0.015847
Rugne Kutta: y1(x) = 0.015847
error:  1.02785870093047e-11
Adams: y2(x) = 0.015847
error:  2.49710488398058e-10
--------------------------------
x = 0.060000
y(x) = 0.018823
Rugne Kutta: y1(x) = 0.018823
error:  1.21681067999369e-11
Adams: y2(x) = 0.018823
error:  3.71658620546000e-10
```

```
y(x) = 0.035381
Rugne Kutta: y1(x) = 0.035381
error:  2.25428772870906e-11
Adams: y2(x) = 0.035381
error:  1.04034147269116e-9
-------------------------------
x = 0.130000
y(x) = 0.037929
Rugne Kutta: y1(x) = 0.037929
error:  2.41301006953520e-11
Adams: y2(x) = 0.037929
error:  1.14209960933032e-9
-------------------------------
x = 0.140000
y(x) = 0.040419
Rugne Kutta: y1(x) = 0.040419
error:  2.56812765497827e-11
Adams: y2(x) = 0.040419
error:  1.24141220914087e-9
-------------------------------
x = 0.150000
y(x) = 0.042849
Rugne Kutta: y1(x) = 0.042849
error:  2.71979452848292e-11
Adams: y2(x) = 0.042849
error:  1.33838621435567e-9
-------------------------------
x = 0.160000
y(x) = 0.045222
Rugne Kutta: y1(x) = 0.045222
error:  2.86818555017554e-11
Adams: y2(x) = 0.045222
error:  1.43312273159779e-9
-------------------------------
x = 0.170000
y(x) = 0.047537
Rugne Kutta: y1(x) = 0.047537
error:  3.01341521180554e-11
Adams: y2(x) = 0.047537
error:  1.52571816292024e-9
-------------------------------
x = 0.180000
y(x) = 0.049795
Rugne Kutta: y1(x) = 0.049795
error:  3.15562020958282e-11
Adams: y2(x) = 0.049795
error:  1.61626369926671e-9
-------------------------------
x = 0.190000
y(x) = 0.051997
Rugne Kutta: y1(x) = 0.051997
error:  3.29496013806718e-11
Adams: y2(x) = 0.051997
error:  1.70484561190509e-9
-------------------------------
x = 0.200000
y(x) = 0.054143
Rugne Kutta: y1(x) = 0.054143
error:  3.43152173343242e-11
Adams: y2(x) = 0.054143
error:  1.79154648061175e-9
-------------------------------
x = 0.210000
y(x) = 0.056233
Rugne Kutta: y1(x) = 0.056233
error:  3.56540283408258e-11
Adams: y2(x) = 0.056233
error:  1.87644461774328e-9
```

```
--------------------------------
x = 0.220000
y(x) = 0.058268
Rugne Kutta: y1(x) = 0.058268
error:  3.69673319733366e-11
Adams: y2(x) = 0.058268
error:  1.95961410293100e-9
--------------------------------
x = 0.230000
y(x) = 0.060248
Rugne Kutta: y1(x) = 0.060248
error:  3.82562315159873e-11
Adams: y2(x) = 0.060248
error:  2.04112550472590e-9
--------------------------------
x = 0.240000
y(x) = 0.062175
Rugne Kutta: y1(x) = 0.062175
error:  3.95214139192746e-11
Adams: y2(x) = 0.062175
error:  2.12104632468790e-9
--------------------------------
x = 0.250000
y(x) = 0.064047
Rugne Kutta: y1(x) = 0.064047
error:  4.07641143063131e-11
Adams: y2(x) = 0.064047
error:  2.19944024104635e-9
--------------------------------
x = 0.260000
y(x) = 0.065866
Rugne Kutta: y1(x) = 0.065866
error:  4.19851514665837e-11
Adams: y2(x) = 0.065866
error:  2.27636820504529e-9
--------------------------------
x = 0.270000
y(x) = 0.067632
Rugne Kutta: y1(x) = 0.067632
error:  4.31852470450522e-11
Adams: y2(x) = 0.067632
error:  2.35188836461564e-9
--------------------------------
x = 0.280000
y(x) = 0.069346
Rugne Kutta: y1(x) = 0.069346
error:  4.43654696313800e-11
Adams: y2(x) = 0.069346
error:  2.42605574518606e-9
--------------------------------
x = 0.290000
y(x) = 0.071008
Rugne Kutta: y1(x) = 0.071008
error:  4.55265408705330e-11
Adams: y2(x) = 0.071008
error:  2.49892308235022e-9
--------------------------------
x = 0.300000
y(x) = 0.072618
Rugne Kutta: y1(x) = 0.072618
error:  4.66686966849039e-11
Adams: y2(x) = 0.072618
error:  2.57054112717814e-9
--------------------------------
x = 0.310000
y(x) = 0.074176
Rugne Kutta: y1(x) = 0.074176
error:  4.77932970976980e-11
```

```
Adams: y2(x) = 0.074176
error:  2.64095716129287e-9
------------------------------
x = 0.320000
y(x) = 0.075684
Rugne Kutta: y1(x) = 0.075684
error:  4.89006057868835e-11
Adams: y2(x) = 0.075684
error:  2.71021734221666e-9
------------------------------
x = 0.330000
y(x) = 0.077141
Rugne Kutta: y1(x) = 0.077141
error:  4.99911362306094e-11
Adams: y2(x) = 0.077141
error:  2.77836545437005e-9
------------------------------
x = 0.340000
y(x) = 0.078548
Rugne Kutta: y1(x) = 0.078548
error:  5.10659847741124e-11
Adams: y2(x) = 0.078548
error:  2.84544274253840e-9
------------------------------
x = 0.350000
y(x) = 0.079906
Rugne Kutta: y1(x) = 0.079906
error:  5.21255955066025e-11
Adams: y2(x) = 0.079906
error:  2.91148917475059e-9
------------------------------
x = 0.360000
y(x) = 0.081214
Rugne Kutta: y1(x) = 0.081214
error:  5.31703570061381e-11
Adams: y2(x) = 0.081214
error:  2.97654301206762e-9
------------------------------
x = 0.370000
y(x) = 0.082473
Rugne Kutta: y1(x) = 0.082473
error:  5.42010186732611e-11
Adams: y2(x) = 0.082473
error:  3.04064046163788e-9
------------------------------
x = 0.380000
y(x) = 0.083683
Rugne Kutta: y1(x) = 0.083683
error:  5.52176776524860e-11
Adams: y2(x) = 0.083683
error:  3.10381671753124e-9
------------------------------
x = 0.390000
y(x) = 0.084845
Rugne Kutta: y1(x) = 0.084845
error:  5.62210694665666e-11
Adams: y2(x) = 0.084845
error:  3.16610486439384e-9
------------------------------
x = 0.400000
y(x) = 0.085959
Rugne Kutta: y1(x) = 0.085959
error:  5.72118880048933e-11
Adams: y2(x) = 0.085959
error:  3.22753652970409e-9
------------------------------
x = 0.410000
y(x) = 0.087026
```

```
Rugne Kutta: y1(x) = 0.087026
error:  5.81903553120711e-11
Adams: y2(x) = 0.087026
error:  3.28814245276199e-9
------------------------------
x = 0.420000
y(x) = 0.088045
Rugne Kutta: y1(x) = 0.088045
error:  5.91568183327951e-11
Adams: y2(x) = 0.088045
error:  3.34795191569981e-9
------------------------------
x = 0.430000
y(x) = 0.089017
Rugne Kutta: y1(x) = 0.089017
error:  6.01118876897289e-11
Adams: y2(x) = 0.089017
error:  3.40699272960432e-9
------------------------------
x = 0.440000
y(x) = 0.089943
Rugne Kutta: y1(x) = 0.089943
error:  6.10559658387189e-11
Adams: y2(x) = 0.089943
error:  3.46529166472820e-9
------------------------------
x = 0.450000
y(x) = 0.090822
Rugne Kutta: y1(x) = 0.090822
error:  6.19892748243700e-11
Adams: y2(x) = 0.090822
error:  3.52287458926792e-9
------------------------------
x = 0.460000
y(x) = 0.091655
Rugne Kutta: y1(x) = 0.091655
error:  6.29120366912872e-11
Adams: y2(x) = 0.091655
error:  3.57976627507473e-9
------------------------------
x = 0.470000
y(x) = 0.092443
Rugne Kutta: y1(x) = 0.092443
error:  6.38249592066487e-11
Adams: y2(x) = 0.092443
error:  3.63598999519876e-9
------------------------------
x = 0.480000
y(x) = 0.093185
Rugne Kutta: y1(x) = 0.093185
error:  6.47283893151496e-11
Adams: y2(x) = 0.093185
error:  3.69156839818974e-9
------------------------------
x = 0.490000
y(x) = 0.093882
Rugne Kutta: y1(x) = 0.093882
error:  6.56221466055484e-11
Adams: y2(x) = 0.093882
error:  3.74652373014150e-9
------------------------------
x = 0.500000
y(x) = 0.094535
Rugne Kutta: y1(x) = 0.094535
error:  6.65070498673259e-11
Adams: y2(x) = 0.094535
error:  3.80087633589099e-9
------------------------------
```

```
x = 0.510000
y(x) = 0.095143
Rugne Kutta: y1(x) = 0.095143
error:  6.73833211450869e-11
Adams: y2(x) = 0.095143
error:  3.85464635210830e-9
------------------------------
x = 0.520000
y(x) = 0.095707
Rugne Kutta: y1(x) = 0.095707
error:  6.82510992167096e-11
Adams: y2(x) = 0.095707
error:  3.90785316606301e-9
------------------------------
x = 0.530000
y(x) = 0.096227
Rugne Kutta: y1(x) = 0.096227
error:  6.91105506156475e-11
Adams: y2(x) = 0.096227
error:  3.96051536011299e-9
------------------------------
x = 0.540000
y(x) = 0.096703
Rugne Kutta: y1(x) = 0.096703
error:  6.99621471866863e-11
Adams: y2(x) = 0.096703
error:  4.01265048965982e-9
------------------------------
x = 0.550000
y(x) = 0.097137
Rugne Kutta: y1(x) = 0.097137
error:  7.08065550636405e-11
Adams: y2(x) = 0.097137
error:  4.06427523580444e-9
------------------------------
x = 0.560000
y(x) = 0.097527
Rugne Kutta: y1(x) = 0.097527
error:  7.16428444347272e-11
Adams: y2(x) = 0.097527
error:  4.11540716782621e-9
------------------------------
x = 0.570000
y(x) = 0.097874
Rugne Kutta: y1(x) = 0.097874
error:  7.24723892009393e-11
Adams: y2(x) = 0.097874
error:  4.16606092679128e-9
------------------------------
x = 0.580000
y(x) = 0.098179
Rugne Kutta: y1(x) = 0.098179
error:  7.32948146620060e-11
Adams: y2(x) = 0.098179
error:  4.21625230562217e-9
------------------------------
x = 0.590000
y(x) = 0.098442
Rugne Kutta: y1(x) = 0.098442
error:  7.41106204182884e-11
Adams: y2(x) = 0.098442
error:  4.26599561231811e-9
------------------------------
x = 0.600000
y(x) = 0.098663
Rugne Kutta: y1(x) = 0.098663
error:  7.49197925919987e-11
Adams: y2(x) = 0.098663
```

```
error:  4.31530514100054e-9
-------------------------------
x = 0.610000
y(x) = 0.098842
Rugne Kutta: y1(x) = 0.098842
error:  7.57223173053490e-11
Adams: y2(x) = 0.098842
error:  4.36419463067939e-9
-------------------------------
x = 0.620000
y(x) = 0.098980
Rugne Kutta: y1(x) = 0.098980
error:  7.65191798812737e-11
Adams: y2(x) = 0.098980
error:  4.41267630768571e-9
-------------------------------
x = 0.630000
y(x) = 0.099076
Rugne Kutta: y1(x) = 0.099076
error:  7.73100472528654e-11
Adams: y2(x) = 0.099076
error:  4.46076324489564e-9
-------------------------------
x = 0.640000
y(x) = 0.099132
Rugne Kutta: y1(x) = 0.099132
error:  7.80951275869413e-11
Adams: y2(x) = 0.099132
error:  4.50846750210676e-9
-------------------------------
x = 0.650000
y(x) = 0.099147
Rugne Kutta: y1(x) = 0.099147
error:  7.88744070057135e-11
Adams: y2(x) = 0.099147
error:  4.55580091707208e-9
-------------------------------
x = 0.660000
y(x) = 0.099122
Rugne Kutta: y1(x) = 0.099122
error:  7.96485655207846e-11
Adams: y2(x) = 0.099122
error:  4.60277417568822e-9
-------------------------------
x = 0.670000
y(x) = 0.099056
Rugne Kutta: y1(x) = 0.099056
error:  8.04172284318838e-11
Adams: y2(x) = 0.099056
error:  4.64939865774117e-9
-------------------------------
x = 0.680000
y(x) = 0.098951
Rugne Kutta: y1(x) = 0.098951
error:  8.11807149281307e-11
Adams: y2(x) = 0.098951
error:  4.69568460503833e-9
-------------------------------
x = 0.690000
y(x) = 0.098806
Rugne Kutta: y1(x) = 0.098806
error:  8.19396633877645e-11
Adams: y2(x) = 0.098806
error:  4.74164159325330e-9
-------------------------------
x = 0.700000
y(x) = 0.098621
Rugne Kutta: y1(x) = 0.098621
```

```
error:  8.26934631881215e-11
Adams: y2(x) = 0.098621
error:  4.78728008623808e-9
-------------------------------
x = 0.710000
y(x) = 0.098398
Rugne Kutta: y1(x) = 0.098398
error:  8.34427527074411e-11
Adams: y2(x) = 0.098398
error:  4.83260893802129e-9
-------------------------------
x = 0.720000
y(x) = 0.098135
Rugne Kutta: y1(x) = 0.098135
error:  8.41874348012084e-11
Adams: y2(x) = 0.098135
error:  4.87763740508740e-9
-------------------------------
x = 0.730000
y(x) = 0.097833
Rugne Kutta: y1(x) = 0.097833
error:  8.49274678360601e-11
Adams: y2(x) = 0.097833
error:  4.92237438309839e-9
-------------------------------
x = 0.740000
y(x) = 0.097493
Rugne Kutta: y1(x) = 0.097493
error:  8.56635734569622e-11
Adams: y2(x) = 0.097493
error:  4.96682769912660e-9
-------------------------------
x = 0.750000
y(x) = 0.097115
Rugne Kutta: y1(x) = 0.097115
error:  8.63954741081585e-11
Adams: y2(x) = 0.097115
error:  5.01100586025593e-9
-------------------------------
x = 0.760000
y(x) = 0.096698
Rugne Kutta: y1(x) = 0.096698
error:  8.71230310117710e-11
Adams: y2(x) = 0.096698
error:  5.05491698499227e-9
-------------------------------
x = 0.770000
y(x) = 0.096244
Rugne Kutta: y1(x) = 0.096244
error:  8.78468825460388e-11
Adams: y2(x) = 0.096244
error:  5.09856810937404e-9
-------------------------------
x = 0.780000
y(x) = 0.095752
Rugne Kutta: y1(x) = 0.095752
error:  8.85670842221131e-11
Adams: y2(x) = 0.095752
error:  5.14196663026212e-9
-------------------------------
x = 0.790000
y(x) = 0.095223
Rugne Kutta: y1(x) = 0.095223
error:  8.92832613397232e-11
Adams: y2(x) = 0.095223
error:  5.18512009717309e-9
-------------------------------
x = 0.800000
```

```
y(x) = 0.094656
Rugne Kutta: y1(x) = 0.094656
error:  8.99959412548057e-11
Adams: y2(x) = 0.094656
error:  5.22803493552271e-9
-------------------------------
x = 0.810000
y(x) = 0.094052
Rugne Kutta: y1(x) = 0.094052
error:  9.07053043786021e-11
Adams: y2(x) = 0.094052
error:  5.27071766787124e-9
-------------------------------
x = 0.820000
y(x) = 0.093412
Rugne Kutta: y1(x) = 0.093412
error:  9.14108511107514e-11
Adams: y2(x) = 0.093412
error:  5.31317527474595e-9
-------------------------------
x = 0.830000
y(x) = 0.092735
Rugne Kutta: y1(x) = 0.092735
error:  9.21134279963098e-11
Adams: y2(x) = 0.092735
error:  5.35541321011745e-9
-------------------------------
x = 0.840000
y(x) = 0.092021
Rugne Kutta: y1(x) = 0.092021
error:  9.28125354349163e-11
Adams: y2(x) = 0.092021
error:  5.39743802430159e-9
-------------------------------
x = 0.850000
y(x) = 0.091271
Rugne Kutta: y1(x) = 0.091271
error:  9.35087007825075e-11
Adams: y2(x) = 0.091271
error:  5.43925507412446e-9
-------------------------------
x = 0.860000
y(x) = 0.090485
Rugne Kutta: y1(x) = 0.090485
error:  9.42017158722663e-11
Adams: y2(x) = 0.090485
error:  5.48087025764588e-9
-------------------------------
x = 0.870000
y(x) = 0.089663
Rugne Kutta: y1(x) = 0.089663
error:  9.48918582599489e-11
Adams: y2(x) = 0.089663
error:  5.52228877903627e-9
-------------------------------
x = 0.880000
y(x) = 0.088806
Rugne Kutta: y1(x) = 0.088806
error:  9.55787116119211e-11
Adams: y2(x) = 0.088806
error:  5.56351638369978e-9
-------------------------------
x = 0.890000
y(x) = 0.087913
Rugne Kutta: y1(x) = 0.087913
error:  9.62632334955416e-11
Adams: y2(x) = 0.087913
error:  5.60455727660614e-9
```

```
--------------------------------
x = 0.900000
y(x) = 0.086985
Rugne Kutta: y1(x) = 0.086985
error:  9.69444524656637e-11
Adams: y2(x) = 0.086985
error:  5.64541742520408e-9
--------------------------------
x = 0.910000
y(x) = 0.086022
Rugne Kutta: y1(x) = 0.086022
error:  9.76233677230098e-11
Adams: y2(x) = 0.086022
error:  5.68610064588526e-9
--------------------------------
x = 0.920000
y(x) = 0.085024
Rugne Kutta: y1(x) = 0.085024
error:  9.82997988563383e-11
Adams: y2(x) = 0.085024
error:  5.72661180975320e-9
--------------------------------
x = 0.930000
y(x) = 0.083991
Rugne Kutta: y1(x) = 0.083991
error:  9.89735515766199e-11
Adams: y2(x) = 0.083991
error:  5.76695566301133e-9
--------------------------------
x = 0.940000
y(x) = 0.082924
Rugne Kutta: y1(x) = 0.082924
error:  9.96447785395205e-11
Adams: y2(x) = 0.082924
error:  5.80713643838493e-9
--------------------------------
x = 0.950000
y(x) = 0.081822
Rugne Kutta: y1(x) = 0.081822
error:  1.00313465867252e-10
Adams: y2(x) = 0.081822
error:  5.84715839635486e-9
--------------------------------
x = 0.960000
y(x) = 0.080686
Rugne Kutta: y1(x) = 0.080686
error:  1.00979918871147e-10
Adams: y2(x) = 0.080686
error:  5.88702538106833e-9
--------------------------------
x = 0.970000
y(x) = 0.079516
Rugne Kutta: y1(x) = 0.079516
error:  1.01644165306780e-10
Adams: y2(x) = 0.079516
error:  5.92674134769489e-9
--------------------------------
x = 0.980000
y(x) = 0.078312
Rugne Kutta: y1(x) = 0.078312
error:  1.02306038640698e-10
Adams: y2(x) = 0.078312
error:  5.96631036242634e-9
--------------------------------
x = 0.990000
y(x) = 0.077075
Rugne Kutta: y1(x) = 0.077075
error:  1.02965705406355e-10
```

```
Adams: y2(x) = 0.077075
error:  6.00573600573195e-9
------------------------------
x = 1.000000
y(x) = 0.075804
Rugne Kutta: y1(x) = 0.075804
error:  1.03623387648355e-10
Adams: y2(x) = 0.075804
error:  6.04502171930310e-9
------------------------------
```

Start building the plots

```
def Dh(h,x0=0,step=1,C=3):
    ############################
    f = lambda x,y: (3*y-2*x+1)/(3*x+3)
    y = lambda x: ((1.0/3)*(x+1)*(C-2*np.log(x+1))-1)
    x = np.arange(x0, x0+step+h, step=h)
    ############################
    ############################
    ########Runge-Kutta########
    y_ = [y(x0)]

    def y_next(x_n,x_next,y_n,h=h,f=f):
        F1 = f(x_n,y_n)
        F2 = f(x_n+h/2.,y_n+h/2.*F1)
        F3 = f(x_n+h/2.,y_n+h/2.*F2)
        F4 = f(x_next,y_n+h*F3)
        return y_n+h/6.*(F1+2*F2+2*F3+F4)
    for n in xrange(x.size-1):
        y_.append(y_next(x[n], x[n+1], y_[-1]))
    ############################
    ##########Adams###########
    def Adams_n(y,x,h=h,f=f):
        return y[-1]+h/24.*(55*f(x[-1],y[-1]) - 59*f(x[-2],y[-2]) + 37*f(x[-3],y[-3]) - 9*f(x[-4],y[-4]))
    y_a = y_[0:4]
    for i in range(4,x.size):
        y_a.append(Adams_n(y_a,x[0:i]))
    ############################
    return [np.abs(np.subtract(y_,y(x))).max(), np.abs(np.subtract(y_a,y(x))).max()]
from decimal import Decimal
h_array=[]
inp=raw_input("Enter n: ")
h_temp=1./float(inp)
print "The step is: ",h_temp
h_cumul=h_temp
while h_cumul<1:
    if not (Decimal('1')%Decimal(str(h_cumul))):
        h_array.append(float(format(h_cumul, '.6f')))
    h_cumul+=h_temp
if h_array[-1]==1.:
    del h_array[-1]
#cleanse h_array
for i in xrange(len(h_array)):
    if h_array[i]>=0.1:
        del h_array[i:]
        break
print h_array
#create a
a = np.array([Dh(h=x) for x in h_array])
```

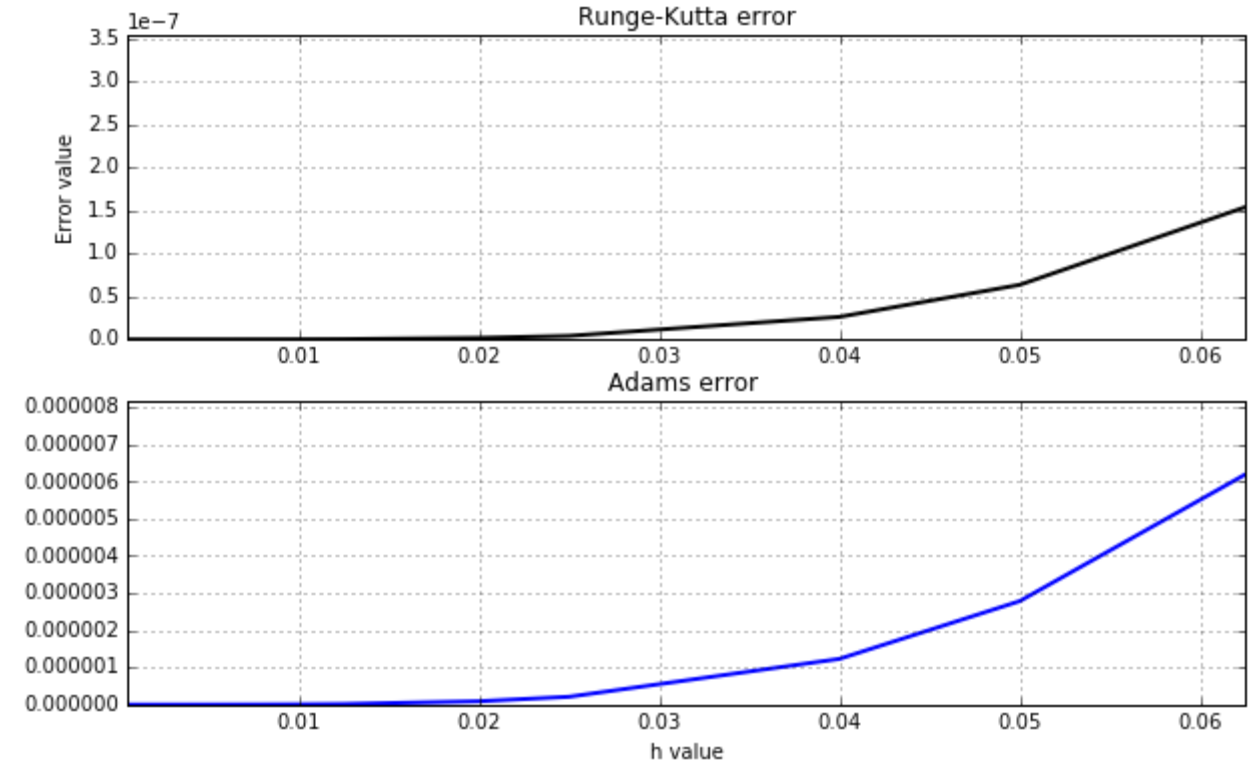
```
Enter n: 2000
```

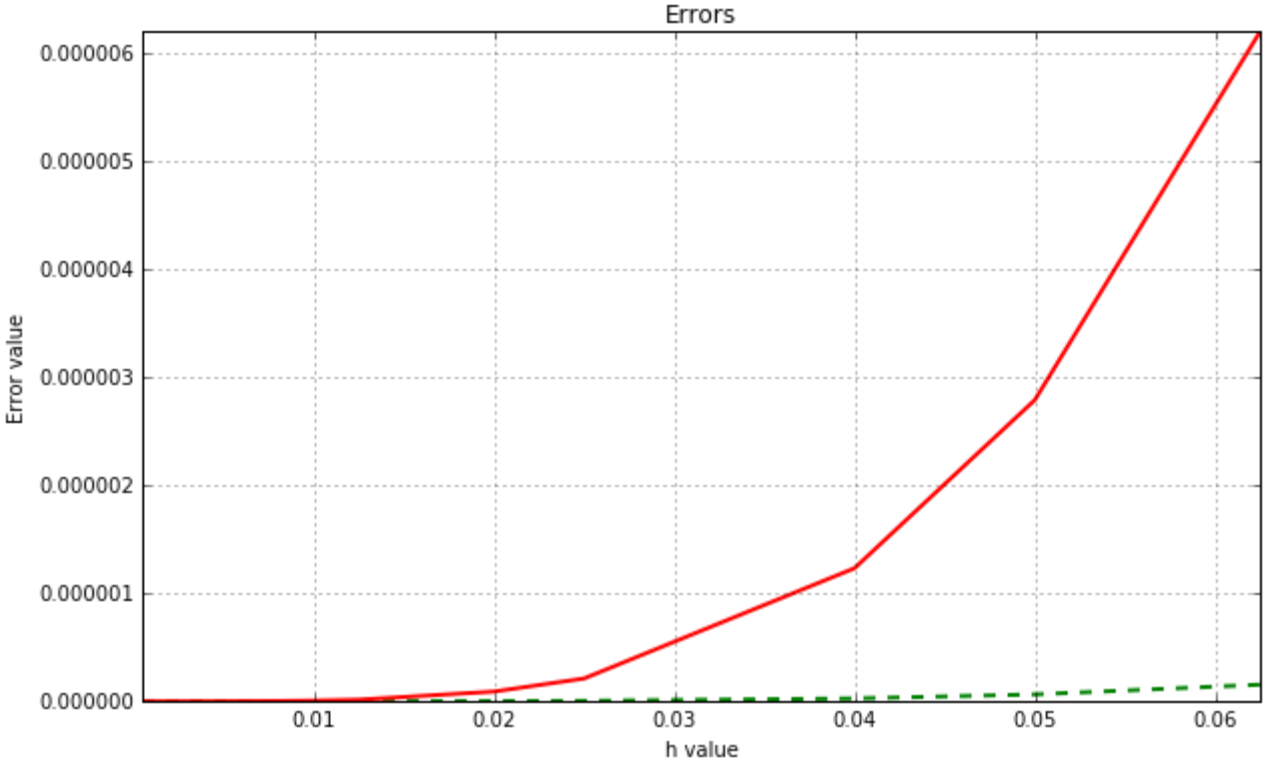
```
The step is:  0.0005
[0.0005, 0.001, 0.002, 0.0025, 0.004, 0.005, 0.008, 0.01, 0.0125, 0.02, 0.025, 0.04,
0.05, 0.0625]
```

```
print "    Runge-Kutta         Adams\n"
for i in xrange(len(a)):
   print a[i], "while h =",h_array[i]
```

```
    Runge-Kutta            Adams

[  8.32667268e-16   4.05647738e-14] while h = 0.0005
[  1.05332409e-14   6.48453513e-13] while h = 0.001
[  1.66713865e-13   1.02953063e-11] while h = 0.002
[  4.06508160e-13   2.50365562e-11] while h = 0.0025
[  2.66116296e-12   1.62163546e-10] while h = 0.004
[  6.49362508e-12   3.92823135e-10] while h = 0.005
[  4.24889707e-11   2.51485630e-09] while h = 0.008
[  1.03623388e-10   6.04502172e-09] while h = 0.01
[  2.52651136e-10   1.44749104e-08] while h = 0.0125
[  1.64911756e-09   8.95300494e-08] while h = 0.02
[  4.01521172e-09   2.10360493e-07] while h = 0.025
[  2.60950629e-08   1.23009701e-06] while h = 0.04
[  6.33449748e-08   2.78520694e-06] while h = 0.05
[  1.53522165e-07   6.19155562e-06] while h = 0.0625
```

```
#Errors (separately)
plt.figure(1, figsize=(10,6))
plt.subplot(2,1,1)
plt.plot(h_array, a[:,0], 'black', linewidth=2.0)
plt.ylabel('Error value')
plt.xlim([h_array[0], h_array[-1]])
plt.ylim([0, (a[:,0]).max()+0.0000002])
plt.title('Runge-Kutta error')
plt.grid(True)

plt.subplot(2,1,2)
plt.plot(h_array, a[:,1], linewidth=2.0)
plt.xlabel('h value')
plt.xlim([h_array[0], h_array[-1]])
plt.ylim([0, (a[:,1]).max()+0.000002])
plt.title('Adams error ')
plt.grid(True)
plt.show()

#Errors (together)
plt.figure(2, figsize=(10,6))
lines = plt.plot(h_array, a[:,0], 'g--', h_array, a[:,1], 'r', lw=2.0)
plt.ylabel('Error value')
plt.xlabel('h value')
plt.xlim([h_array[0], h_array[-1]])
maxs=[(a[:,0]).max(), (a[:,1]).max()]
plt.ylim([0, max(maxs[0], maxs[1]) + 0.000002])
plt.title('Errors')
plt.text(0.004, .000025, r'$GREEN - RungeKutta,\ RED - Adams$', fontsize=18, color='black')
plt.grid(True)
```

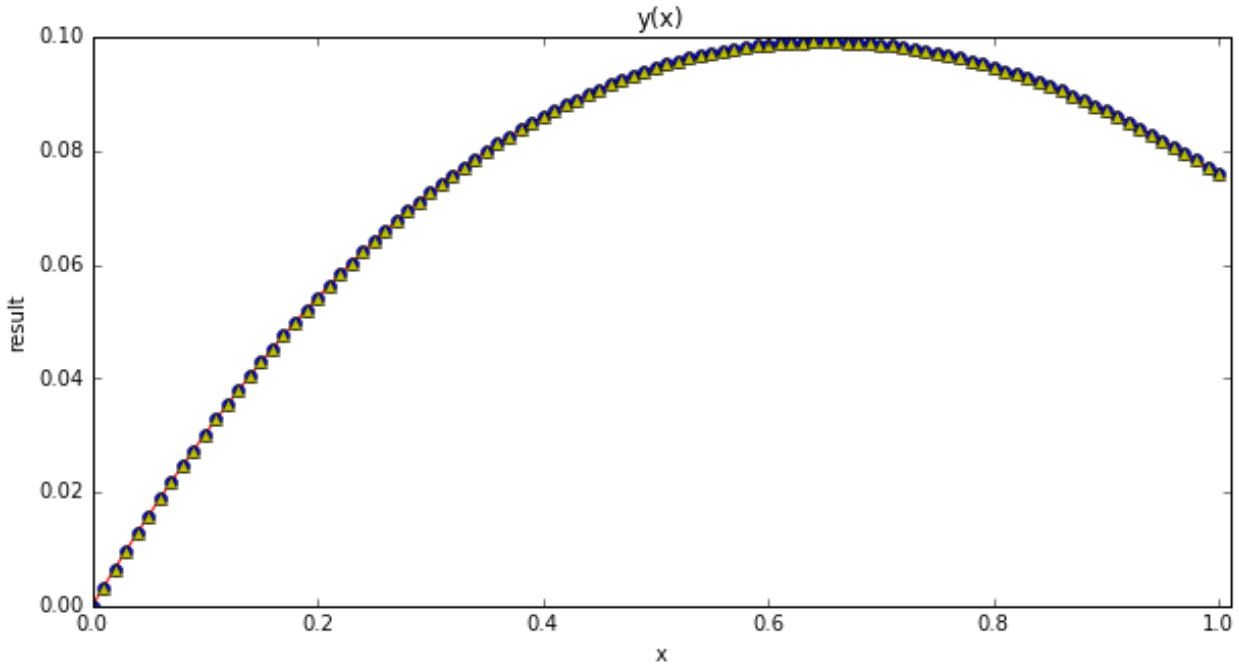




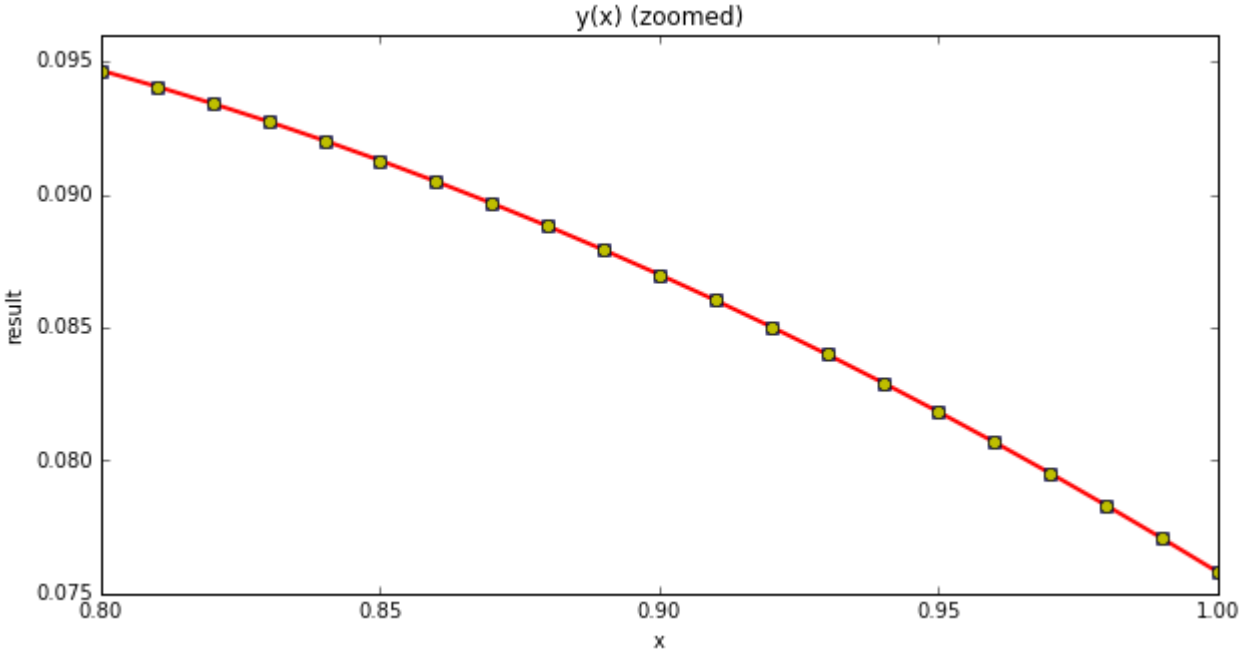
$GREEN - RungeKutta,\ RED - Adams$

```
x = np.arange(x0, x0+1+h, step=h)
plt.figure(1, figsize=(10,5))
plt.xlim([x0,x0+1+h])
plt.plot(x,y_y,'r', x, y_a,'bo', x, y_, 'y^')
plt.title('y(x)')
plt.xlabel('x')
plt.ylabel('result')
plt.text(0.1, -0.02, r'$RED - strict\ values,\ BLUE - Adams,\ YELLOW-RungeKutta$', fontsize=16,
color='black')
plt.figure(2, figsize=(10,5))
plt.xlim([0.8, 1])
plt.ylim([0.075, 0.096])
```

```
plt.plot(x,y_y,'r', x, y_a,'bs', x, y_, 'yo', lw=2.0)
plt.title('y(x) (zoomed)')
plt.xlabel('x')
plt.ylabel('result')
plt.show()
```



$$RED - strict\ values,\ BLUE - Adams,\ YELLOW - RungeKutta$$



Висновки:
графіки помилок для обох методів нагадують параболи четвертого порядку. Метод Рунге-Кута виявився більш точним, до того ж він потребує лише одну початкову точку. Також з графіків видно, що чим менше крок — тим точніше працюють дані методи.